Convex Optimization of Resource Allocation in Asymmetric and Heterogeneous MultiCores

Amir Morad, Leonid Yavits and Ran Ginosar

Abstract— Chip area, power consumption, execution time, off-chip memory bandwidth, overall cache miss rate and Network on Chip (NoC) capacity are limiting the scalability of Chip Multiprocessors (CMP). Consider a workload comprising a sequential and multiple concurrent tasks and asymmetric or heterogeneous multicore architecture. A convex optimization framework is proposed, for selecting the optimal set of processing cores and allocating area and power resources among them, the NoC and the last level cache, under constrained total area, total average power, total execution time and off-chip bandwidth. The framework relies on analytical performance and power models of the processing cores, NoC and last level cache as a function of their allocated resources. Due to practical implementation of the cores, the optimal architecture under constraints may exclude several of the cores. Several asymmetric and heterogeneous multi-core configurations are explored. Convex optimization of multicores is shown to extend optimizations that are based on Lagrange multipliers. We find that our framework obtains the optimal chip resources allocation over a wide spectrum of parameters and constraints, and thus can automate complex architectural design, analysis and verification.

Index Terms—Chip Multiprocessors, Modeling of computer architecture

1 Introduction

With the growth of the number of transistors that can be integrated into a single silicon die, coupled with the growth of the available number heterogeneous multicore building blocks available to the chip architect, finding the optimal architecture of a large scale multicore processor under rigid physical constraints such as area, power and available off-chip bandwidth, is extremely complex and time consuming. The chip architect must select, for example, the optimal number of processing cores to integrate, the task allocation among the cores, the cache hierarchy configuration, the Network on a Chip (NoC) topology, and the resource allocation (e.g., area, power) among the hardware building blocks. In doing so, the chip architect must take into account the performance of each of the building blocks, as a function of the resources it consumes.

While a workload characterized by high task-level parallelism may lead the architect to select a symmetric multicore architecture, a high data-parallelism workload may tip the balance towards SIMD machines. The closer we approach the maximal power density the chip package may disperse, the likelihood of finding multiple smaller sized heterogeneous cores [28] (for example, DCT engine, graphics accelerator etc.) increases. To that end, analytical models for most building blocks of modern ICs (e.g., caches, NoC, processing units) have been researched, enabling the exploration of the chip design space in a reasonable timeframe.

This work focuses on optimization of a comprehensive multiprocessor architecture containing several heterogeneous building blocks, using convex optimization methodology [34], for a given workload, under constrained resources. The contributions of this work are:

(a) Formulation and solution of three convex optimization problems, for finding the optimal:

- Execution time under total area, total average power and off-chip bandwidth constraints; or,
- Average power under total area, execution time and off-chip bandwidth constraints; or,
- Chip area under execution time, total average power and off-chip bandwidth constraints.

(b) Extending the framework defined by [40], [7] and [6] by:

- Considering several multicore building blocks, not just the processing elements; and,
- Considering a workload containing both sequential and concurrent sections, as opposed to a series of either sequential or concurrent tasks; and,
- Detailing the exact optimal resource allocation, rather than merely providing the necessary condition for optimality.

The rest of this paper is organized as follows: Section 2 presents and discusses relevant related work. Section 3 proposes and investigates analytical models for common multicore building blocks. Section 4 describes a convex optimization framework, and exemplifies it by deriving the optimal execution time, total average power, and total area

- *Amir Morad (*), E-mail: amirm@tx.technion.ac.il.*
- *Leonid Yavits (*), E-mail: yavits@tx.technion.ac.il.*
- *Ran Ginosar (*), E-mail: ran@ee.technion.ac.il.*

(*) Authors are with the Department of Electrical Engineering, Technion-Israel Institute of Technology, Haifa 32000, Israel.

of a resource constrained asymmetric and heterogeneous multicore processors. Section 6 summarizes and concludes the paper.

2 Related work

Analytical models of common building blocks of modern ICs have been thoroughly studied. Polack [14] modeled the performance of modern CPUs as a square root function of the resource assigned to them. Liwei *et al.* [41] presented an analytical access time model for on-chip cache memories that shows the dependence of cache access time on cache parameters. Wilton *et al.* [36] described an analytical model for the access and cycle times of direct-mapped and set-associative caches. Tsai *et al.* [43] explored the architectural design of cache memories using 3D circuits. Muralimanohar *et al.* [29], [30] modeled different types of wires, such as RC-modeled wires with different power/delay characteristics and differential low-swing buses and Non-uniform Cache Access (NUCA). Krishna *et al.* [5] researched the effects of data sharing in multi-threaded applications on optimal area allocation between cores and cache. Yavits *et al.* [19] developed an analytical model for cache hierarchy levels and described the optimal allocation of constrained resources among them.

A substantial body of literature explores NoC topologies and optimization. W. Liwei *et al.* proposed a buffer allocation algorithm for wormhole routing networks-on-chip. Ben-Itzhak *et al.* [42] modeled the delay of a wormhole routing based NoC with variable link capacities and a variable number of virtual channels per link. Z. Guz *et al.* [45] introduce Nahalal, a non-uniform cache (NUCA) topology that enables fast access to shared data for all processors, while preserving the vicinity of private data to each processor.

Optimization framework consolidating common multicore building blocks have been extensively studied as well: Cassidy *et al.* [3] have optimized processor area, L2 cache area and the number of cores for an area-constrained symmetric multicore using Lagrange multipliers [31]. Oh *at el.* [37] presented an analytical model to study the trade-off of the core count and the cache capacity in a CMP under a finite die area constraint, suggesting optimization of L1 / L2 cache sizes by varying the division of a constrained cache area among them, and evaluating each division's effect on the resulting performance of the CMP. Alameldeen *at el.* [2] used analytical modeling to study the trade-off between the number of CMP cores and cache size. Wentzlaff *at el.* [13] introduced an analytic model to study the tradeoffs of utilizing increased chip area for larger caches versus more cores, focusing on manycore architectures. Huh *at el.* [17] compared the area and performance tradeoffs for CMP implementations to determine how many processing cores future server CMPs should have, whether the cores should have in-order or out-of-order issues, and how big the per-processor on-chip caches should be. Zhao *et al.* [21] developed a constraint-aware analysis methodology considering total chip area and bandwidth limitations. Morad *et al* [39] and Hill *et al* [22] augmented Amdahl's law with a corollary to multicore architecture by constructing a model for multicore performance and speedup. These models assumed that the parallelizable portion of the workload could be equally scaled by any number of cores (implying that tasks can break up indefinitely, which is seldom the case), and the optimal resource allocation among the cores was not explored. To that end, S. Natarajan *et al* [35] created an elaborated model that evaluates the execution time of both the serial part and the parallel part of the application, taking into account the scaling of both these execution times with the input problem size and the number of processors. Concluding that symmetric many cores using very simple cores will only be able to achieve very high performance on a very specialized class of applications.

Analytical models for power/energy optimization is a well-studied subject: Elyada *et al.* [4] have considered a multiprocessor with unknown workload and attempted to dynamically set frequency-voltage work-points for each core, with a goal to minimize a defined energy-performance criterion. Rotem *et al.* [4] studied the optimal voltage and frequency operational point of the processor in order to achieve minimum energy of the computing platform.

The interactions between multiple parallel processors incur performance overheads. These overheads are a result of synchronization, communication and coherence costs. Analytical models for these overheads have been studied as well. Morad *et al.* [39] modeled the synchronization, communication and coherence as a time penalty on Amdahl law, concluding that asymmetric multiprocessors can reduce power consumption by more than two thirds with similar performance compared to symmetric multiprocessors. Yavits *et al.* [20] studied the overheads and concluded that in applications with high inter-core communication requirements, the workload should be executed on a small number of cores, and applications of high sequential-to-parallel synchronization requirements may better be executed by the sequential core. Lau *et al.* [15] introduced an extension to Hill and Marty's multi-core cost/performance model to account for the uncore components, concluding that to sustain the scalability of future many-core systems, the uncore components must be designed to scale sub-linearly with respect to the overall core count.

Zaidenberg *et al.* [40] introduced MultiAmdahl, a resource constrained optimization framework for optimal resource allocation in CMP. Morad *et al.* [7] have presented the Generalized MultiAmdahl model (Figure 1) that minimized sequential execution time of a heterogeneous set of accelerators by optimally selecting which accelerators to allocate and what area to assign to the allocated accelerators.

While the Generalized MultiAmdahl addresses a series of heterogeneous workload segments that are not concurrent (only one core operates at a time), Morad *et al.* [6] presented a framework that, given a multicore and a workload consisting of concurrent tasks and resource constraints

(Figure 2), finds the optimal selection of a subset of the available cores and the optimal resource allocation among them.

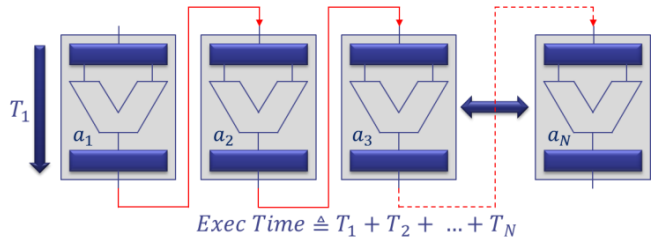

Figure 1. Generalized MultiAmdahl: Optimizing multicore resource allocation executing sequential series of heterogeneous workload segments

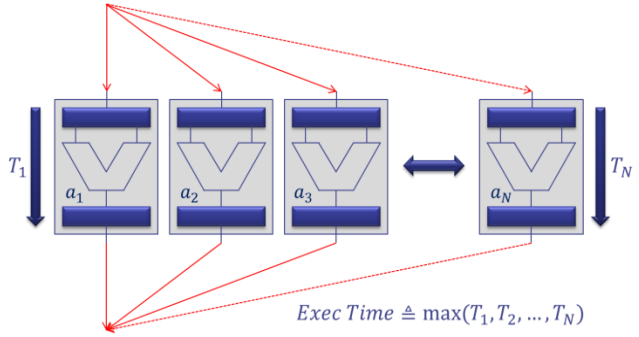

Figure 2. Optimizing multicore resource allocation executing concurrent series of workload segments

The limitations of the frameworks presented in [40], [7] and[6] is: (a) modeling the processing cores, but not addressing common building blocks such as NoC and LLC; (a) modeling workloads containing either a sequence of sequential heterogeneous tasks ([40], [7](, or modeling workloads containing a sequence of concurrent sections ([5]), but not both together; (c) utilizing Lagrange multipliers thus identifying the necessary condition for optimality, but not the optimal point; and (d) modeling constrained area ([40], [7](, or constrained area/power designs ([5]), but not addressing off-chip bandwidth; and (e) solving for optimal execution time under area/power constraints, but not addressing optimizing power or area under constraints.

When optimizing a modern multicore, the chip architect may need to optimize workloads containing both sequential and concurrent sections, and take into account additional building blocks such as caches and NoC while adhering to off-chip bandwidth limitations.

In this paper we address the following research question: given (a) a multicore architecture consisting of last level cache (LLC), processing cores and a NoC interconnecting the cores and the LLC (see Figure 3); (b) workloads consisting of sequential and concurrent tasks (see Figure 4); and (c) physical resource constraints (area, power, execution time, off-chip bandwidth), what is the optimal selection of a subset of the available processing cores and what is the optimal resource allocation among all blocks.

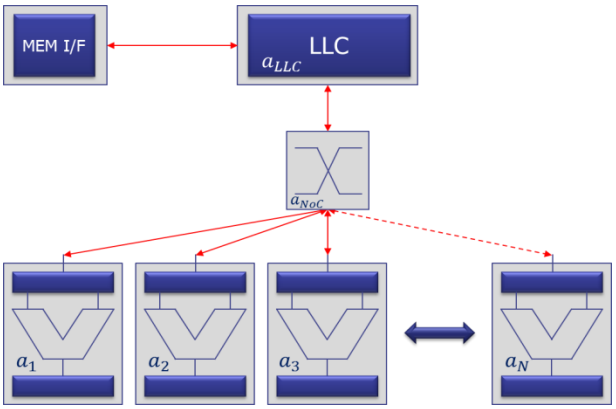

Figure 3. Typical multicore architecture containing several processing cores, LLC, NoC and Memory I/F.

At the optimal resource allocation point, all blocks reach an equilibrium state in which if some area is taken from one block and given to another, the overall optimization target (e.g., area, power, execution time) increases. While searching for such optimal point, one must take into account all block' characteristics, total area, power, execution time and off-chip bandwidth constraints and workload specifics. Thus, any *a-priori*, *ad-hoc* selection criteria based on performance threshold or otherwise (let alone universal criteria) electing a subset of the cores may yield only sub-optimal results. Hence, the key contribution of this paper is to enable the multicore architect to select an optimal subset of the cores and allocate resources among the cores, the LLC and the NoC, without having to explore the design space in an iterative *ad-hoc* fashion.

3 Analytical Models

In this section, we propose and investigate an analytical model for common building blocks of modern multi-processors. For each block, we model the delay (e.g., execution time, access time) and power, as a function of the resources it uses (e.g., area).

3.1 Workload

Consider a typical workload consisting of sequential as well as M concurrent tasks, as depicted in Figure 3. Further, consider a multicore processor comprising N cores, each capable of executing the following:

- *Sequential (un-parallelizable) section*: The sequential portion of the workload cannot break into a finer granularity concurrent sub-tasks, and thus it is executed on a single core. The sequential section of the workload requires t_0 seconds to execute on a *reference processor* of IPS=1 (Instructions Per Second). The sequential section can be accelerated by any one of N available cores. The performance of core j is $Perf_j(a_j)$, relative to the performance on the reference processor, and is a function of its area a_j. The acceleration function $f_j(a_j)$ represents the inverted performance of core j :

$$f_j(a_j) = \frac{1}{Perf_j(a_j)} \qquad (1)$$

The runtime of the sequential task running on core j having area a_j is thus $f_j(a_j) \cdot t_0$.

- *Concurrent section*: We assume that the concurrent portion of the workload is composed of M tasks, and each task may run on any core. In a similar manner to sequential processing, each concurrent task i of the workload requires time t_i to execute on a *reference processor* of IPS=1. The runtime of the i^{th} task running on core j having area a_j is thus $f_j(a_j) \cdot t_i$.
- *Sequential - parallel synchronization*: Concurrency incurs data exchange between the sequential (first) core and the other cores at the beginning and the end of each concurrent section of the workload for subsequent processing. The data exchange entails transferring data from the LLC through the NoC into the private caches of the cores.

The performance of a core increases when additional area resources are assigned to it. Therefore, the acceleration functions $f_j(a_j)$ are strictly decreasing. We assume that the acceleration functions $f_j(a_j)$ are convex and are continuously differentiable.

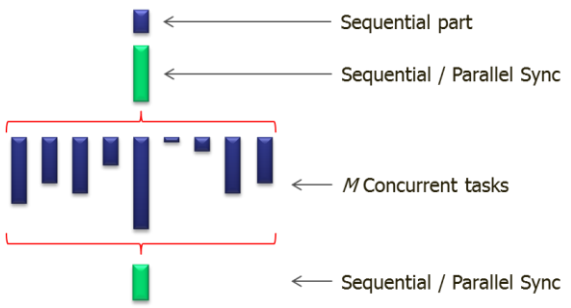

Figure 4. Typical workload consisting of sequential and concurrent tasks

3.2 Processing Core

The following analysis and definitions is based on [6]. Following Pollack's rule [14] and according to [39], the processing core's inverted performance may be written as follows (coefficients translating from area to performance units are scaled to unity):

$$f_{Core}(a_{Core}) = \frac{1}{a_{Core}^{\beta}} \tag{2}$$

The exponent β typically varies from 0.3 to 0.7 [33]. The higher the β, the stronger the core is. For the purpose of our optimization framework, following [6], we modify the acceleration function (1) of core j to depend on both its area a_j and its dynamic power p_j, as follows:

$$f_j(a_j, p_j) = \frac{1}{Perf_j(a_j, p_j)} \tag{3}$$

Note that p_j represents actual power dissipated in core j rather than the maximum power that can be consumed by that core, as determined by its area, maximum voltage, maximum frequency and other physical constraints; for instance, it is possible that at some optimum point for the entire chip, a particular core is operated at $p_j < P_{\max j}$.

Area is a static resource, i.e., it does not change during execution. Power, however, has both static and dynamic components, and the distribution of these resources among the cores is not identical. Static power depends on temperature [18], and temperature, in turn, depends on power density (related to dynamic power). We shall separate the static power consumption of the core into two components, idle static power, and temperature induced static power.

Assume that the multicore employs Dynamic Voltage and Frequency Scaling (DVFS). The idle static power is annotated as a manufacturing technology related constant s_1 multiplied by the core's allocated area a_j representing the static power when: (a) gate temperature is at the low end of the operating conditions range; (b) core voltage is at the low end of the DVFS voltage range; and, (c) core frequency is at the low end of the DVFS frequency range. Within the normal operating temperature range (say, 55°C–85°C), leakage power consumption may be estimated using a linear function of temperature [44]. Thus in normal operating conditions, we can express the temperature induced static power as a manufacturing technology related constant s_2 times its dynamic power p_j.

$$P_{Proc-Static} = \sum_{j=1}^{j=N} s_1 a_j + s_2 p_j \tag{4}$$

Note that in our analysis we choose to ignore floorplan-induced leakage power, that is, temperature increase due to the heat generated by adjacent cores. Given that the cores have large enough radial shape, floorplan induced leakage is limited to the boundary and hence is a second order effect on total leakage. Note however that this assumption may not always hold.

Given that the multicore employs DVFS. Each core's frequency corresponds to its power budget, p_j, enabling clocking at a range of frequencies, from zero (when the core is idle) to F_{max} (maximal frequency possible by the operating conditions and the physical constraints of the design). The maximal dynamic power of core j can be written as follows:

$$P_{max\ j} = \alpha_1 C_j F_{\max j} V_{max}^2 \tag{5}$$

where α_1 is the activity factor. Voltage is inversely proportional to gate delay, and thus it is proportional to frequency $V_{\max j} = \alpha_2 F_{\max j}$, where α_2 is a constant translating Hz to Volts. Capacitance C_j is proportional to area a_j, $C_j = \alpha_3 a_j$. Assume that all cores are subject to the same activity factor α_1. Assume further that all cores are driven by the same voltage range ($\forall j, V_{min} \le V_j \le V_{max}$). We can write (5) as:

$$P_{max\ j} = \alpha_1 C_j F_{\max j} \left(\alpha_2 F_{\max j}\right)^2 = \alpha a_j F_{\max j}^3 \tag{6}$$

where α is a constant absorbing α_1, α_2 and α_3. We can also model $P_{max\ j} = c_j a_j$, where $c_j = \alpha F_{\max j}^3$ is a constant translating units of area to power, in agreement with [1] and [26]. Next, assume that each core is driven at some operating frequency, $F_{\text{oper } j}$, dissipating dynamic power p_j. Following (6):

$$p_j = \alpha a_j F_{\text{oper } j}^3 \tag{7}$$

Equation (7) complements [32] who noted through an empirical study that a typical CPU power is a polynomial function of frequency $p_{core} \propto f^{\eta}$ where η typically ranges from 1.5 (in a low power manufacturing technology) to 2.4 (in a high performance manufacturing technology). The present theoretical analysis leads to $\eta = 3$ instead, but any other number will do and will not significantly change our results. We annotate the normalized frequency of the core as $F_{norm\ j}$, ranging from 0 (when the core is idle) to 1 (when the core is driven at maximal frequency corresponding to the operating conditions and the physical constraints of the design). Thus:

$$F_{norm\ j} = \frac{F_{oper\ j}}{F_{max\ j}} = \left(\frac{p_j}{P_{max\ j}}\right)^{\frac{1}{3}} = \left(\frac{p_j}{c_j a_j}\right)^{\frac{1}{3}} \tag{8}$$

Even when assigned with infinite power budget, the core's operating frequency cannot exceed its maximum frequency determined by the physical constraints of the design, thus (8) is revised as follows:

$$F_{norm\ j} = \min\left(\left(\frac{p_j}{c_j a_j}\right)^{\frac{1}{3}}, 1\right) \tag{9}$$

Since the core's inverted performance is modeled as a power law with respect to its allocated area, we can write the acceleration function of a core as follows:

$$f_j(a_j, p_j) = \frac{1}{{a_j}^{\beta_j}} \frac{1}{F_{norm\ j}} = \frac{1}{{a_j}^{\beta_j}} max\left(\left(\frac{c_j a_j}{p_j}\right)^{\frac{1}{3}}, 1\right) \tag{10}$$

Note that if core j is assigned with null power ($p_j = 0$), $f_j \to \infty$. Conversely, if core j is assigned with infinite power ($p_j \to \infty$), $f_j \to {a_j}^{-\beta_j}$.

While considering asymmetric architectures, we focus on homogenous tasks and exclude task heterogeneity (for example, certain tasks may benefit from larger cache sizes, others may benefit from larger branch prediction buffer, etc.). Thus, we assume that tasks runtime depends only on the core's speedup function at its designated area.

Given the core areas $A_{Core} = \{a_1, \dots, a_N\}$, and power allocation $P_{Core} = \{p_1, \dots, p_N\}$, the execution time t_0 of the sequential portion of the workload is determined by the fastest available core, thus:

$$T_{Seq} \triangleq \sum_{j=1}^{N} f_j(a_j, P_{Seq})\, b_{0,j} t_0 \tag{11}$$

where, at the optimal point:

$$b_{i,j} = \begin{cases} 1: & i^{th}\ task\ is\ assigned\ to\ j^{th}\ core \\ 0: & otherwise \end{cases} \tag{12}$$

Note that when the sequential portion of the workload is executed on some core, the dynamic power that is assigned to all other cores becomes available to it (up to its maximum possible power $c_j a_j$), thus:

$$P_{Seq} = P_{Proc-Dynamic} = \sum_{i=1}^{N} p_i$$

Note further that a task is assigned to a single core. The area and power of core j, namely (a_j, p_j), may be 0 or positive, respectively.

Core j execution time of the concurrent tasks assigned to it is:

$$T_j \triangleq \sum_{i=1}^{i=M} f_j(a_j, p_j)\, b_{i,j} t_i = f_j(a_j, p_j) \sum_{i=1}^{i=M} b_{i,j} t_i \tag{13}$$

The total execution time of the concurrent section of the workload T_{Con}, is determined by the last core to finish executing its allocated tasks, thus

$$T_{Con} \triangleq max(T_1, T_2, \dots, T_N) \tag{14}$$

The total execution time is:

$$T_{Proc} = T_{Seq} + T_{Con} \tag{15}$$

The total average power dissipated by the processing cores is thus:

$$P_{Proc} = P_{Proc-Static} + P_{Proc-Dynamic} \tag{16}$$

3.3 Last Level Cache (LLC)

To model the LLC power as a function of its area resources, we follow the circuit diagrams detailed in [36] and [38], and conduct design space exploration using CACTI-6 [29], [30]. The LLC static and dynamic power simulated for 45nm by CACTI-6 are shown in Figure 5, plotted as the red and green graph, respectively. The cache size simulated for 45nm is shown in Figure 6, plotted in the red graph.

While varying the cache size, the per-access dynamic power dissipation of a cache can be approximated by power-law model:

$$P_{LLC-Dynamic}(S_{LLC}) = c_1 + c_2 {S_{LLC}}^{\gamma} \tag{17}$$

where S_{LLC} is the LLC size and is equal to Block Size × Associativity × Number of Sets. With the aid of numerical approximation tools such as [27], the constant c_1, c_2, and the exponent γ are found by fitting the power law (17) curve to the cache dynamic power data, either received by exploring circuit level models or generated by CACTI-6 (see the blue dotted fitted function in Figure 5). Note that the technology node and the selection of the internal cache architecture (e.g., block size, associativity, number of sets) affect these constants. Hence, in our analysis, we have fixed the technology node and cache architecture, and varied the cache size (number of bits allocated), while recording the area and static and dynamic power of each configuration.

Further design space exploration with CACTI-6, yields that the cache size can be approximated by power-law model:

$$S_{LLC} = c_3 + c_4 {A_{LLC}}^{\vartheta} \tag{18}$$

where A_{LLC} is the total area allocated to the cache. Our anal-

ysis shows that ϑ is very close to 1 (see the blue dotted fitted function in Figure 6) thus (17) is re-written as:

$$P_{LLC-Dynamic}(A_{LLC}) = c_1 + c_2(c_3 + c_4 A_{LLC})^{\gamma} \quad (19)$$

The analysis further shows that the static power of the cache can be approximated as a linear function of the cache size, and hence cache area (see the black dotted fitted function in Figure 5):

$$P_{LLC-Static}(A_{LLC}) = c_5 + c_6(c_3 + c_4 A_{LLC}) \quad (20)$$

The total power allocated to the LLC is thus:

$$P_{LLC} = P_{LLC-Static} + P_{LLC-Dynamic} \quad (21)$$

Figure 5 and Figure 6 demonstrate that the power-law, (17) and (18), approximates CACTI simulations (and their underlying circuit level models) over a wide range of cache sizes (from 4Kbytes to 16Mbytes) to within 5%. Note that equation (17) is approximately in agreement with [3] who modeled the cache dynamic power as a square root of its assigned size.

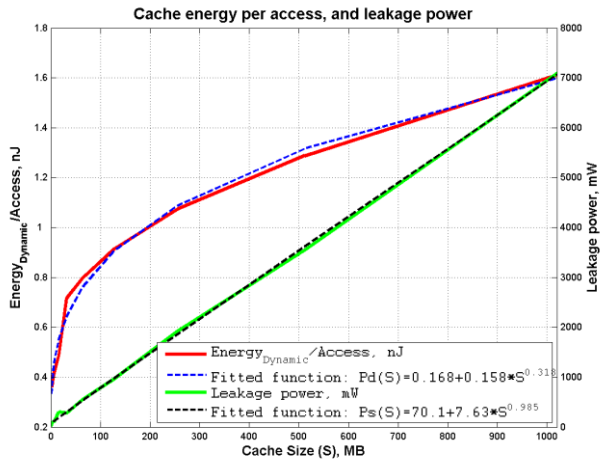

Figure 5. Cache dynamic read energy/access vs. cache size (CACTI-6).

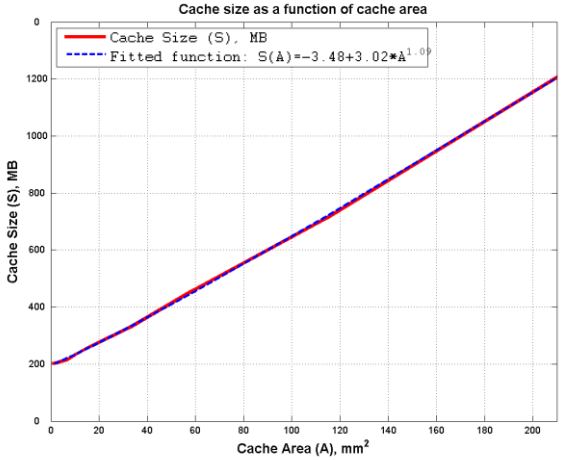

Figure 6. Cache area vs. cache size (CACTI-6).

We assume a typical hierarchical cache configuration, in which each processing core contains a single private cache level, and all cores share the LLC. This framework can be extended to any number of private, shared or hybrid levels. Following [19], we assume that the access time of the LLC is approximated by power-law model:

$$Access_{LLC}(S_{LLC}) = \tau S_{LLC}^{\rho} \quad (22)$$

Both τ and the exponent ρ are found by fitting the power law (22) curve to the cache access time data, either received by exploring circuit level models or generated by CACTI. For caches having several shared clients, the access time can be written as follows:

$$t_{LLC} = T_{NoC} + \tau\left(\frac{S_{LLC}}{N_{Proc}}\right)^{\rho}$$
$$T_{NoC} = t_{transfer} + t_{blocking} + t_{congestion} \quad (23)$$

where N_{Proc} is the number of shared cache clients (in our framework, the number of processing cores). NoC delay T_{NoC} is a sum of transfer delay, $t_{transfer}$, blocking delay, $t_{blocking}$ and queuing (congestion) delay, $t_{congestion}$. [???] We adopt the analytical models for $t_{blocking}$ and $t_{congestion}$ proposed by [41] and [42]. $t_{blocking}$ and $t_{congestion}$ depend on a variety of parameters including the shared cache access rate R_{LLC}, the network capacity, the number of cores etc. Those parameters except for R_{LLC} and the number of cores, are not part of our optimization framework. Therefore we model both $t_{blocking}$ and $t_{congestion}$ as function of R_{LLC}, assuming the rest of parameters are constant. Transfer delay, $t_{transfer}$, is $O(\sqrt{n})$, assuming 2-D mesh NoC [13]. The average memory delay can be written as follows:

$$T_{LLC} = (1 - MR_{\mathrm{LLC}})t_{\mathrm{LLC}} + MR_{LLC}(d_{Access} + d_{Queue}) \quad (24)$$

where d_{Access} is the DRAM access penalty and d_{Queue} is the interconnect queuing delay. The LLC miss-rate, MR_{LLC}, can be written as follows:

$$MR_{LLC} = m_{Comp} + (1 - m_{Comp})C_{Sharing}\Big/\sqrt{\frac{S_{LLC}}{N_{Proc}}} \quad (25)$$

where $C_{Sharing}$ is the data sharing factor [3], and m_{Comp} is the compulsory miss component reflecting access to data originated in remote (rather than in local) core [17]. Note that m_{Comp} does not depend on the size of the processing cores' local cache.

DRAM interconnect queuing delay d_{Queue} can be presented as a function of the rate of access to off-chip MR_{LLC}, the off-chip memory bandwidth, the number of processing cores etc. [5]. Those parameters except for MR_{LLC} and the number of processing cores are not part of our optimization framework. Hence we model d_{Queue} as a function of MR_{LLC}, assuming the rest of the parameters are constant. The off-chip bandwidth can be written as:

$$BW_{LLC} = R_{LLC} MR_{LLC} \quad (26)$$

In real life, cache performance is affected by a combination of constrained chip resources. When LLC area budget is limited, the elevated miss rate causes the off-chip memory traffic to intensify, thus enlarging the total chip power. As a result, the average memory delay is affected by longer DRAM queuing delays. When LLC area budget

is substantial, LLC power consumption becomes the primary constraint as it becomes too large to power, while on the other hand, the decreased bandwidth reduces the total chip power. To that end, the optimal cache size should be obtained only within the framework of the entire chip. We thus include the DRAM pinout power dissipation into the optimization framework as follows:

$$P_{DDR} = c_{DDR} BW_{LLC} \quad (27)$$

where c_{DDR} is a constant translating bandwidth (bit/second) to average power dissipated on the multicore pinout.

3.4 Network on a Chip (NoC)

In this work, we assume a 2-D mesh NoC [13], with a transfer delay $O(\sqrt{n})$. We assume that the NoC hub blocks are identical and consume a fixed area, a_{NoC}. Thus, the total area of the NoC is:

$$A_{NoC} = (N_{Proc} + 1_{LLC}) a_{NoC} \quad (28)$$

where N_{Proc} is the number of integrated processing cores (that is, the sum of processing cores which have been allocated area larger than 0).

The optimization framework can utilize other NoC topology. For example, Nahalal [45] (see Figure 7), a non-uniform cache (NUCA) that enables fast access to shared data for all processors, while preserving the vicinity of private data to each processor may decreases the shared cache access latency by up to 54% compared to traditional CMP designs.

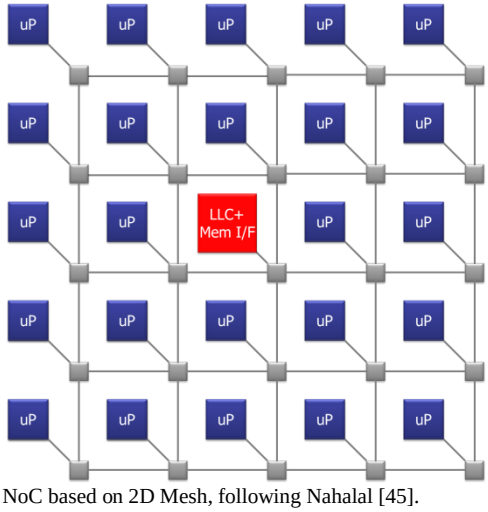

Figure 7. NoC based on 2D Mesh, following Nahalal [45].

We further assume that the NoC blocks are constantly active and thus consume a fixed power, relative to the area:

$$P_{NoC} = C_{AP} A_{NoC} \quad (29)$$

where C_{AP} is a is a constant translating units of area to power [12], [26].

3.5 Total Resources

The combined area, power (static and dynamic) and execution time is a summary of all resources allocated to the processing cores, NoC and the LLC, respectively:

$$A_{Total} = A_{LLC} + A_{NoC} + A_{Proc} \le A_{Total\ Constraint} \quad (30)$$

$$P_{Total} = P_{LLC} + P_{DDR} + P_{NoC} + P_{Proc} \le P_{Total\ Constraint}$$

$$T_{Total} = T_{LLC} + T_{NoC} + T_{Proc} \le T_{Total\ Constraint}$$

$$BW_{Total} = BW_{LLC} \le BW_{Total\ Constraint}$$

In this work, the network delay, T_{NoC}, is modeled as a part of the LLC's delay.

4 Convex Optimization

In this section we propose and investigate an optimization framework based on [34] of constrained area, average power and execution time, asymmetric and heterogeneous multi-core processors.

A convex function satisfies the inequality $f(\alpha x + \beta y) \le \alpha f(x) + \beta f(y)$ [34]. An optimization problem of finding some $x^* \in X$ such that $f(x^*) = min\{f(x): x \in X\}$ where $X \subset \mathbb{R}^n$ is the feasible set and $f(x): \mathbb{R}^n \to \mathbb{R}$ is the objective, is called convex if X is a closed convex set and $f(x)$ is convex on $\mathbb{R}^n$ [16] and [9]. A convex minimization problem consists of the following three parts:

- Objective function: A convex function $f(x): \mathbb{R}^n \to \mathbb{R}$ to be minimized over the variable x.
- Inequality constraints of the form $g_i(x) \le 0$, where the functions g_i are convex.
- Equality constraints of the form $h_i(x) = 0$, where the functions $h_i(x)$ are affine (linear).

In an asymmetric multicore architecture, all cores share the same instruction set architecture (ISA) but may differ in performance, clock frequency, cache size and other micro-architectural characteristics. Heterogeneous architecture, on the other hand, allows different functionality and different ISA in the cores. In an asymmetric multicore architecture, each task can be executed on any one of the cores, whereas in a heterogeneous architecture, tasks can typically execute only on designated cores. We extend the analysis of [6] by considering: (a) comprehensive multicore architecture having NoC and LLC; and (b) workload containing sequential and concurrent tasks, in four types of multicore architectures:

- Asymmetric, unbalanced beta (cf. (2)): the workload may be distributed across up to two distinct cores, having unbalanced exponent $\beta = \{0.6, 0.4\}$, respectively.
- Asymmetric, balanced beta: the workload may be distributed across up to two distinct cores, having balanced exponent $\beta = \{0.5, 0.5\}$, respectively.
- Heterogeneous, unbalanced beta: the workload may be distributed across exactly two distinct cores, having unbalanced exponent $\beta = \{0.6, 0.4\}$, respectively.
- Heterogeneous, balanced beta: the workload may be distributed across exactly two distinct cores, having balanced exponent $\beta = \{0.5, 0.5\}$, respectively.

In this work, we assume that each processing core contains a private L1 cache. Each workload task is characterized by miss rates defined in [5] using PARSEC [10] and NAS [11], corresponding to the sequential-parallel syn-

chronization data exchange. Further, we have applied assumptions similar to those used in [5], [20] and [37] for the constants described in (23) - (25).

Our framework finds the optimal resource allocation in a broad spectrum of constraints. In the following subsections, we exemplify some of these scenarios.

4.1 Optimizing Execution Time under Constraints

Consider a synthetic workload consisting of a sequential task incurring execution time $t_0 = 40$, and two concurrent tasks, incurring execution times $t_c = \{50,10\}$ on a reference processor, respectively. We wish to minimize the total execution time of a multicore consisting of processing cores, NoC and LLC, under total area, total average power and off-chip bandwidth constraints. We thus write our optimization problem as follows:

$$\begin{aligned} &minimize\ T_{Total} \\ &\{A_{LLC,}\ A_{NoC,} a_1, \dots, a_N, p_1, \dots, p_N\} \\ &subject\ to: A_{Total} \le A_{Total\ Constraint} \\ &\qquad P_{Total} \le P_{Total\ Constraint} \\ &\qquad BW_{Total} \le BW_{Total\ Constraint} \end{aligned} \tag{31}$$

To solve problem (31) we used CVX, a package for specifying and solving convex programs [23], [24], replacing the objective functions and the constraints with their convex counterparts.

We optimize the four multicore architectures, as detailed in Figure 8-Figure 10. Each part of the sub-figures depicts three sets of bars corresponding to increasing total areas, $A_{Total-Constraint} = \{5\text{mm}^2, 40\text{mm}^2, 320\text{mm}^2\}$, respectively. Within each set, there are two levels of power budget, high power for the first bar and lower power for the second bar, as follows:

- The optimal area allocation under higher power constraint is depicted in the first bar of each set of Figure 8's sub-figures.
- The optimal area allocation under low average power constraint is depicted in the second bar of each set of Figure 8's sub-figures. Some of the multicore components are power-limited and hence the optimal area allocation is different from areas in the first bar of each set.
- The optimal average power allocation under higher power constraint is depicted in the first bar of each set of Figure 9's sub-figures.
- The optimal average power allocation under low average power constraint is depicted in the second bar of each set of Figure 9's sub-figures. Some of the multicore components are power-limited and hence the optimal power allocation is different from the first bar of each set.
- The red stars in Figure 10's sub-figures depict the optimal execution time corresponding to the higher/lower constrained power scenarios.

The optimal (actually utilized) total area A_{Total}, total power P_{Total}, total execution time T_{Total}, and total bandwidth BW_{Total} are specified at the bottom of each subfigure. Note that for some configurations, the optimal (actually utilized) value may be smaller than its corresponding constraint.

TABLE 1 Summarizes the constraints used for execution time optimization, where SxBy corresponds to set x, bar y in Figure 8-Figure 10.

TABLE 1
EXECUTION TIME OPTIMIZATION CONSTRAINTS

Constraint	S1B1	S1B2	S2B1	S2B2	S3B1	S3B2
$A_{Total\ Constraint}$	5.0	5.0	40.0	40.0	320.0	320.0
$P_{Total\ Constraint}$	1.35	1.08	10.8	8.64	86.40	69.12
$BW_{Total\ Constraint}$	1.9	1.9	1.6	1.6	1.1	1.1

Power constraints are specified in Watt, Execution time constraints in seconds, and Bandwidth constraints in MB/sec.

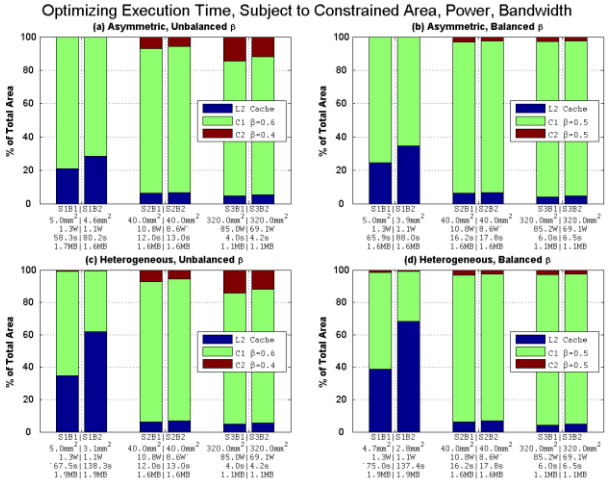

Figure 8. Multicore area partitioning following optimization of execution time, subject to three scenarios of constrained resources.

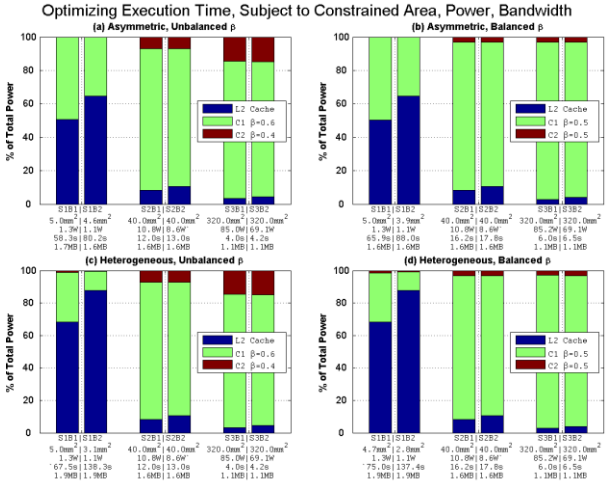

Figure 9. Multicore power partitioning following optimization of execution time, subject to three scenarios of constrained resources.

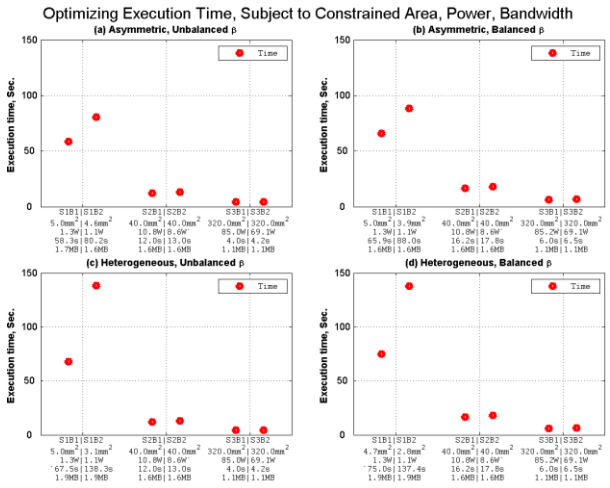

Figure 10. Multicore optimization of execution time, subject to three scenarios of constrained resources.

As shown in Figure 8(a)-Figure 10(a), we consider an asymmetric dual-core architecture where each core has a distinct exponent, $\beta_j = \{0.6, 0.4\}$. Initially, due to constrained area, only a single core is allocated. As the total multicore area increases, the weaker core is assigned with the shorter task, and provided with growing portion of the total average area and power. When power is constrained the weaker core is provided with smaller area portion. On the other hand, the weaker core is provided with larger portion of the total average power at the expense of the stronger cores. We thus find that in a power limited architecture, the optimum is found when area is transferred from the weaker cores to the stronger ones, while power is transferred in the opposite direction, from the stronger cores to the weaker one.

Next, in Figure 8(b)-Figure 10(b) we consider an asymmetric dual-core architecture where all core share the same exponent, $\beta_j = 0.5$. In contrast with the previous set, Figure 8(b)-Figure 10(b) shows that when sufficient area is provided to the multicore such that both cores are allocated, tasks, area and average power allocation do not shift from one core to another even if the area resources increases indefinitely; rather, partitioning among the cores remains constant. Note that although both cores are of the same strength (same β), resources (e.g., area and power) allocation is not equal due to the serial and two concurrent tasks having different execution times.

Next, in Figure 8(c)-Figure 10(c) we consider a heterogeneous dual-core having unbalanced exponents $\beta_j = \{0.6, 0.4\}$, respectively. As the total multicore area gets larger, area resources shift from the stronger core to the weaker core having lower β. Note that in the heterogeneous case all cores must be retained while in the asymmetric multicore some cores may be eliminated. When power is constrained, the stronger core is provided with even larger portion of the total area, at the expense of the weaker core. On the other hand, the weaker core is provided with even larger portion of the total average power. We thus find that in a power-limited architecture, the optimum is found when area is transferred from the weaker cores to the stronger ones, while power is transferred in the opposite direction.

Lastly, in Figure 8(d)-Figure 10(d) we consider a heterogeneous dual-core architecture where all core share the same exponent, $\beta_j = 0.5$. In contrast with the unbalanced heterogeneous multicore, Figure 8(d)-Figure 10(d) show that when sufficient area is provided to the multicore, tasks, area and power allocations do not shift even if the area resources increases indefinitely, similarly to the balanced asymmetric multicore of Figure 8(b)-Figure 10(b).

4.2 Optimizing Power under Constraints

Consider a synthetic workload consisting of a sequential task incurring execution time $t_0 = 40$, and two concurrent tasks, incurring execution times $t_c = \{50,10\}$ on a reference processor, respectively. We wish to minimize the total average power of a multicore consisting of processing cores, NoC and LLC, under total area, total execution time and off-chip bandwidth constraints. We thus write our optimization problem as follows:

$$\begin{aligned} &minimize\ P_{Total} \\ &\{A_{LLC,}\ A_{NoC,} a_1, \dots, a_N, p_1, \dots, p_N\} \\ &subject\ to: A_{Total} \le A_{Total\ Constraint} \\ &\qquad T_{Total} \le T_{Total\ Constraint} \\ &\qquad BW_{Total} \le BW_{Total\ Constraint} \end{aligned} \tag{32}$$

We solve (32) in a similar manner to (31). We optimize the four multicore architectures, as detailed in Figure 11-Figure 13. Each part of the sub-figure depicts three sets of bars corresponding to increasing total areas, $A_{Total\ Const} = \{8\text{mm}^2, 9\text{mm}^2, 10\text{mm}^2\}$, respectively. Within each set, there are two levels of execution time budget, short execution time for the first bar (40s), and long execution time for the second bar (50s), as follows:

- The optimal area allocation under short execution time constraint is depicted in the first bar of each set of Figure 11's sub-figures.
- The optimal area allocation under long execution time constraint is depicted in the second bar of each set of Figure 11's sub-figures.
- The optimal average power allocation under short execution time constraint is depicted in the first bar of each set of Figure 12's sub-figures.
- The optimal average power allocation under long execution time constraint is depicted in the second bar of each set of Figure 12's sub-figures.
- The red stars in Figure 13's sub-figures depict the optimal consumed average power.

The optimal (actually utilized) total area A_{Total}, total power P_{Total}, total execution time T_{Total}, and total bandwidth BW_{Total} are specified at the bottom of each subfigure. Note that for some configurations, the optimal (actually utilized) value may be smaller than its corresponding constraint.

TABLE 2 Summarizes the constraints used for Power optimization, where SxBy corresponds to set x, bar y in Figure 11(a)-Figure 13.

TABLE 2
POWER OPTIMIZATION CONSTRAINTS

Constraint	S1B1	S1B2	S2B1	S2B2	S3B1	S3B2
$A_{Total\ Constraint}$	8.00	8.00	9.00	9.00	10.00	10.00
$T_{Total\ Constraint}$	40.00	50.00	40.00	50.00	40.00	50.00
$BW_{Total\ Constraint}$	2.0	2.0	1.9	1.9	1.9	1.9

Power constraints are specified in Watt, Execution time constraints in seconds, and Bandwidth constraints in MB/sec.

As shown in Figure 11(a)-Figure 13(a), we consider an asymmetric dual-core architecture where each core has a distinct exponent, $\beta_j = \{0.6, 0.4\}$. The optimal power is achieved when the stronger core having the larger exponent, is solely deployed. With a long execution time constrain, the LLC is provided with even larger area and average power portion, at the expense of the stronger core. As the total multicore area increases, with short execution time constraint, the LLC's and the stronger core's area and power portions remains somewhat constant, while the total power decreases. With long execution time constraint, the optimal architecture consumed less than the budgeted total chip area (budget was 9mm^2 and 10mm^2 respectively, and actual consumed area was 8.3mm^2).

Next, in Figure 11(b)-Figure 13(b) we consider an asymmetric dual-core architecture where all core share the same exponent, $\beta_j = 0.5$. In contrast with the previous set, Figure 11(b)-Figure 13(b) shows that with a short execution time, dual processing cores are deployed, while with a long execution time constraint only a single core is deployed. Note further that in the second and third sets, when a long execution time constraint is used, the optimal architecture consumed less than the allowed total chip area (budget was 9mm^2 and 10mm^2 respectively, and actual consumed area was 8.7mm^2).

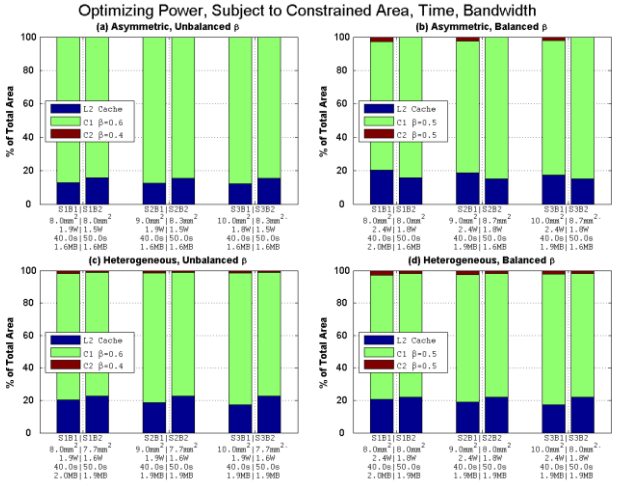

Figure 11. Multicore area partitioning following optimization of total average power, subject to three scenarios of constrained resources.

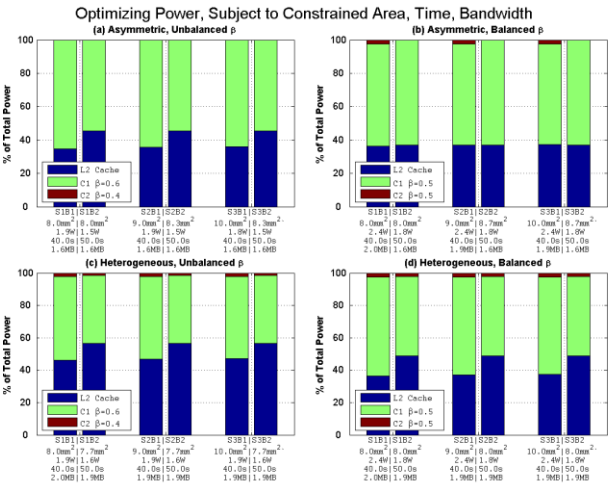

Figure 12. Multicore power partitioning following optimization of total average power, subject to three scenarios of constrained resources.

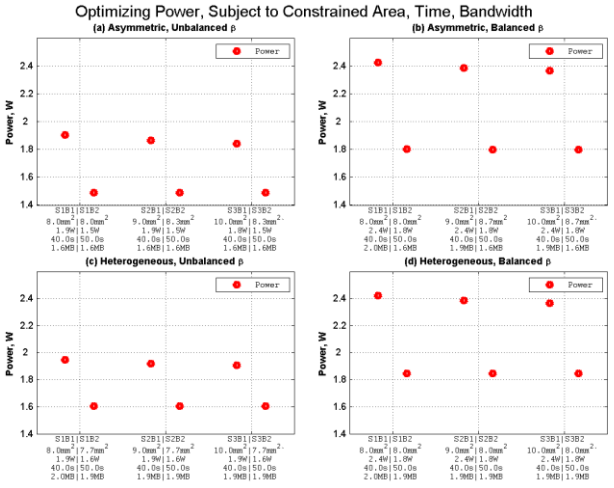

Figure 13. Multicore optimization of total average power, subject to three scenarios of constrained resources.

Next, in Figure 11(c)-Figure 13(c), we consider a heterogeneous dual-core having unbalanced exponents $\beta_j = \{0.6, 0.4\}$, respectively. The optimal average power is achieved when both cores are deployed. With long execution time constrain, the LLC is provided with even larger area and average power portion, at the expense of the processing cores. As the total multicore area increases when short execution time constraint is used, the LLC's and the processing cores' allocated area and power portions remains somewhat constant, while the optimal power decreases. With long execution time constraint, the optimal architecture consumed less than the allowed total chip area (budget was 9mm^2 and 10mm^2 respectively, and actual consumed area was 7.7mm^2). Since in the heterogeneous multicore all processing cores must be retained, the LLC's area almost doubled to compensate for the reduced per core miss rate when compared with the single core implementation in Figure 11(a)-Figure 13(a).

Lastly, in Figure 11(d)-Figure 13(d) we consider a heterogeneous dual-core architecture where all core share the same exponent, $\beta_j = 0.5$. In contrast with the unbalanced heterogeneous multicore, Figure 11(c)-Figure 13(c) shows that area

and average power allocations do not change, but remain the same as in the first set. With a long execution time constraint the optimal architecture consumed less than the allowed total chip area (budget was $8mm^2$, $9mm^2$ and $10mm^2$ respectively, and actual consumed area was $8mm^2$).

4.3 Optimizing Area under Constraints

Consider a synthesized workload consisting of a sequential task incurring execution time $t_0 = 40$, and two concurrent tasks, incurring execution times $t_c = \{50,10\}$ on a reference processor, respectively. We wish to minimize the total area of a multicore consisting of processing cores, NoC and LLC, under total average power, total execution time and off-chip bandwidth constraints. We thus write our optimization problem as follows:

$$\begin{aligned} &minimize\ A_{Total} \\ &\{A_{LLC,}\ A_{NoC,} a_1, \dots, a_N, p_1, \dots, p_N\} \\ &subject\ to: T_{Total} \le T_{Total\ Constraint} \\ &\qquad P_{Total} \le P_{Total\ Constraint} \\ &\qquad BW_{Total} \le BW_{Total\ Constraint} \end{aligned} \tag{33}$$

We solve (33) in a similar manner to (31) and (32). We optimize the four multicore architectures, as detailed in Figure 14-Figure 16. Each sub-figure depicts three sets of bars, each set corresponding to three multicore increasing total average power constraint, $P_{Total\ Const} = \{2.4W,\ 2.5W,\ 2.6W\}$, respectively. Within each set, we varied the execution time budget from short (40s) for the first bar, to long (50s) for the second bar, as follows:

- The optimal area allocation under short execution time constraint is depicted in the first bar of each set of Figure 14's sub-figures.
- The optimal area allocation of under long execution time constraint is depicted in the second bar of each set of Figure 14's sub-figures.
- The optimal average power allocation under short execution time constraint is depicted in the first bar of each set of Figure 15's sub-figures.
- The optimal average power allocation of under long execution time constraint is depicted in the second bar of each set of Figure 15's sub-figures.
- The red stars in Figure 16's sub-figures depict the optimal consumed area.

The optimal (actually utilized) total area A_{Total}, total power P_{Total}, total execution time T_{Total}, and total bandwidth BW_{Total} are specified at the bottom of each subfigure. Note that for some configurations, the optimal (actually utilized) value may be smaller than its corresponding constraint.

TABLE 3 summarizes the constraints used for area optimization, where SxBy corresponds to set x, bar y in Figure 14-Figure 16.

TABLE 3
AREA OPTIMIZATION CONSTRAINTS

Constraint	S1B1	S1B2	S2B1	S2B2	S3B1	S3B2
$P_{Total\ Constraint}$	2.40	2.40	2.50	2.50	2.60	2.60
$T_{Total\ Constraint}$	40.00	50.00	40.00	50.00	40.00	50.00
$BW_{Total\ Constraint}$	4.2	4.2	4.2	4.2	4.2	3.29

Power constraints are specified in Watt, Execution time constraints in seconds, and Bandwidth constraints in MB/sec.

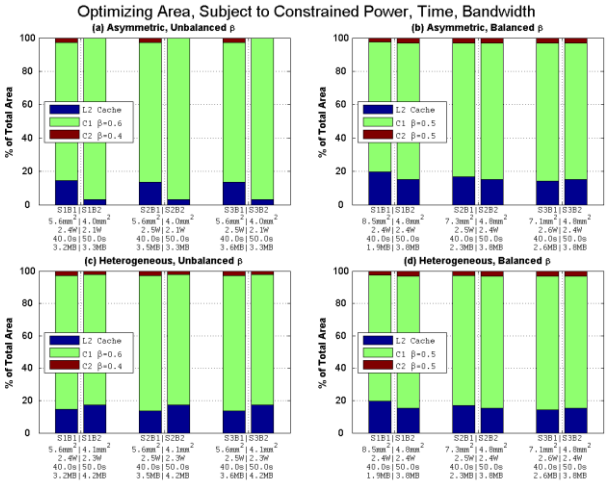

Figure 14. Multicore area partitioning following optimization of total area, subject to three scenarios of constrained resources.

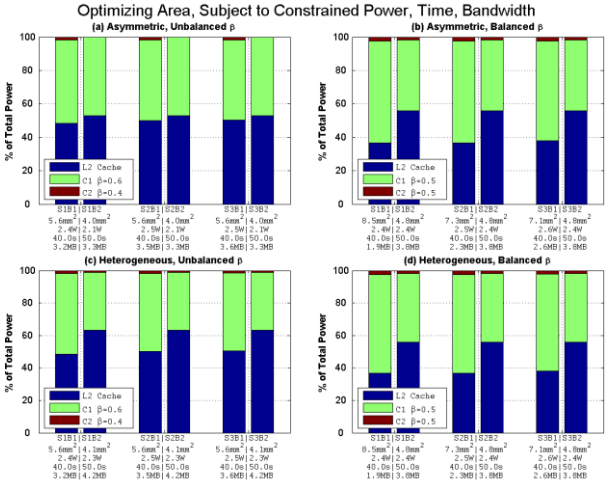

Figure 15. Multicore power partitioning following optimization of total area, subject to three scenarios of constrained resources.

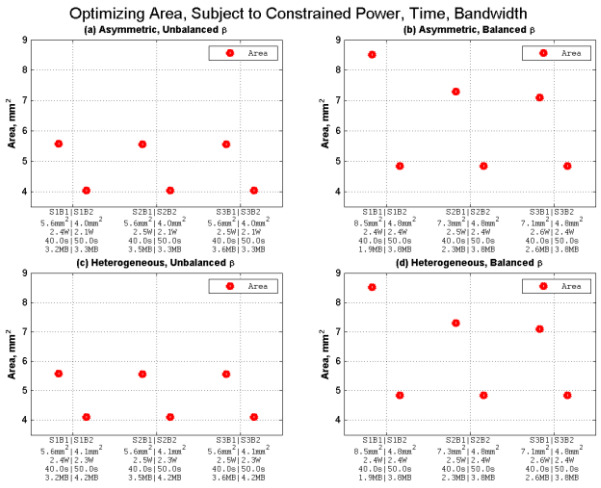

Figure 16. Multicore optimization of total area, subject to three scenarios of constrained resources.

As shown in Figure 14(a)-Figure 16(a), we consider an asymmetric dual-core architecture where each core has a distinct exponent, $\beta_j = \{0.6, 0.4\}$. With a short execution time constrain, dual cores are deployed, however, with a long execution time constrain, the weaker core is eliminated (processing time is no longer the limiting factor) and the LLC is reduced substantially (miss rate is no longer the limiting factor). Larger total average power budget does not improve the optimal area. Note that in the second and third set, when long execution time constraint is used, the optimal architecture consumed less than the budgeted total chip power (budget was 2.5W and 2.6W respectively, and actual consumed power was 2.1W).

Next, in Figure 14(b)-Figure 16(b) we consider an asymmetric dual-core architecture where all core share the same exponent, $\beta_j = 0.5$. In contrast with the previous set, Figure 14(b)-Figure 16(b) shows that with a long execution time constrain, dual processing cores are deployed, and the LLC's area is reduced while its power increases due to miss-rate causing escalated off-chip bandwidth. When the execution time is shortened, area and power resources are taken from the processing units and transferred to the LLC. Note that in the second and third sets, when a long execution time constraint is used, the optimal architecture consumed less than the budgeted total chip total average power (budget was 2.5W and 2.6W respectively, and actual consumed power was 2.4W).

Next, in Figure 14(c)-Figure 16(c), we consider a heterogeneous dual-core having unbalanced exponents $\beta_j = \{0.6,\ 0.4\}$, respectively. With long execution time, the LLC is provided with even larger area and average power portion (when compared with the shorter constraint), at the expense of the processing cores. As the total multicore average power budget increases, the LLC's and the processing cores' allocated area and power portions remains constant since the optimal area of the second set, is also applicable to the third set. Note that with a long execution time constraint, the optimal power consumption was less than the budgeted amount (budget was 2.4W, 2.5W and 2.6W respectively, and actual average power consumption was 2.3W).

Lastly, in Figure 14(d)-Figure 16(d) we consider a heterogeneous dual-core architecture where all core share the same exponent, $\beta_j = 0.5$. In contrast with the unbalanced heterogeneous multicore, Figure 14(d)-Figure 16(d) show that with a short execution time constraint, the optimal chip area reduces as the average power budget grows. However, with a long execution time constraint, the optimal architecture power consumption similar to the one in the first set, and less than the budgeted amount (budget was 2.4W, 2.5W and 2.6W respectively, and actual average power consumption was 2.4W).

5 Discussion

The optimization framework described in Section 7 leads to consistent results, in spite of parameter and constraint variability. Note the similarities among Figure 8, Figure 11 and Figure 14, and also the similarities of Figure 9, Figure 12 and Figure 15. There is no coherent hand-rule conclusion that can guide architects on how to manually partition the chip given a workload and a set of constraints. The reason is that the processing cores, the network, last-level cache and off-chip bandwidth are so intertwined and dependent on each other, that taking some area (or power) from one block and allocating it to the other is likely to impact the allocation of all other blocks. On the other hand, we argue that our framework does find the optimal partitioning over a wide spectrum of parameters and constraints and thus can automate complex architectural design, analysis and verification. We thus conclude that in the multi-core era, when the number of transistors (and thus the number and complexity of functional blocks) grows exponentially, chip architects should embrace automated frameworks for partitioning chip resources, instead of conventional manual methods.

6 Conclusions

This paper describes a multi-core optimization framework that, given (a) workload consisting of sequential and concurrent parts, (b) multicore building blocks models such as processing cores, LLC and NoC, (c) constrained area and average power budget, (d) limited off-chip memory bandwidth, and (e) limited NoC capacity, utilizes convex optimization to find an optimal subset of the processing cores and allocates area and power resources among all, whether optimizing for, e.g., area, power or execution time. The framework relies on modeling the performance of each core as a function of its area and power. To that end, models are provided and analyzed for core power and performance, NoC power and delay, cache miss rate and power and off-chip bandwidth, all as functions of area.

We show that there is no coherent hand-rule conclusion that can guide architects on how to manually partition the chip given a workload and a set of constraints. On the other hand, we argue that our framework does find the optimal

partitioning over a wide spectrum of parameters and constraints and thus can automate complex architectural design, analysis and verification. We conclude that chip architects should embrace automated frameworks for partitioning resources among the chip, instead of conventional manual methods.

This paper extends and generalizes previous Lagrange based optimization frameworks [40][7] and [6]. One of the potential applications of this framework is optimizing the average power or energy of a given workload, executing on a fabricated semiconductor chip (where area resources have been already partitioned). In such a case, for example, a real-time workload is given with known execution time upper bounds, and the average power budget is optimally partitioned so as to minimize total average power or energy. Our framework offers an efficient alternative to iterative ad-hoc methods for exploring the design space.

7 Acknowledgment

This research was funded in part by the *Intel Collaborative Research Institute for Computational Intelligence* (ICRI-CI) and by Hasso-Plattner Institute (HPI).

8 References

[1] A. Agarwal et al. "Core Count vs. Cache Size for Manycore Architectures in the Cloud." (2010).

[2] A. Alameldeen, "Using compression to improve chip multiprocessor performance", *PhD thesis, University of Wisconsin, Madison*, WI, 2006.

[3] A. Cassidy and A. Andreou, "Beyond Amdahl Law - An objective function that links performance gains to delay and energy", *IEEE Transactions on Computers*, vol. 61, no. 8, pp. 1110-1126, Aug 2012.

[4] A. Elyada, R. Ginosar, U. Weiser, "Low-Complexity Policies for Energy-Performance Tradeoff in Chip-Multi-Processors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Volume: 16, Issue: 9, 2008, Page(s): 1243 - 1248.

[5] A. Krishna, A. Samih, and Y. Solihin. "Data sharing in multi-threaded applications and its impact on chip design", *ISPASS*, 2012.

[6] A. Morad, T. Morad, L. Yavits and R. Ginosar, "Optimization of Asymmetric and Heterogeneous MultiCore," 2013

[7] A. Morad, T. Morad, L. Yavits, R. Ginosar, U. C. Weiser. "Generalized MultiAmdahl: Optimization of Heterogeneous Multi-Accelerator SoC," *IEEE Computer Architecture Letters*, 2012.

[8] A. S. Cassidy and A. G. Andreou. "Beyond Amdahl's law: An objective function that links multiprocessor performance gains to delay and energy," IEEE Transactions on Computers, 2011.

[9] Ben-Tal, Aharon; Nemirovskiĭ, Arkadiĭ Semenovich (2001). Lectures on modern convex optimization: analysis, algorithms, and engineering applications. pp. 335–336.

[10] C. Bienia *et al.*, "The PARSEC Benchmark Suite: Characterization and Architectural Implications", *PACT* 2008..

[11] D. Bailey, *et al.* "The NAS Parallel Benchmarks", *Intl. Journal of Supercomputer Applications*, 5(3):63–73, 1991

[12] D. H. Woo and H. H. S. Lee. "*Extending Amdahl's law for energy-efficient computing in the many-core era.*" Computer, 41(12):24–31, 2008.

[13] D. Wentzlaff, *et al*., "Core Count vs. Cache Size for Manycore Architectures in the Cloud." *Tech. Rep. MIT-CSAIL-TR-2010-008*, MIT, 2010.

[14] F. Pollack. "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies," *Keynote, Micro 32*, 1999, www.intel.com/research/mrl/Library/micro32Keynote.pdf

[15] G. Loh, "The cost of uncore in throughput-oriented many-core processors", *Workshop on Architectures and Languages for Throughput Applications*. 2008.

[16] Hiriart-Urruty, Jean-Baptiste; Lemaréchal, Claude (1996). Convex analysis and minimization algorithms: Fundamentals. p. 291.

[17] Huh *et al*., "Exploring the design space of future CMPs," *PACT*, 2001.

[18] K. Banerjee et al., "A self-consistent junction temperature estimation methodology for nanometer scale ICs with implications for performance and thermal management," IEEE IEDM, 2003, pp. 887-890.

[19] L. Yavits, A. Morad, R. Ginosar, "Cache Hierarchy Optimization," *IEEE Computer Architecture Letters*, 2013

[20] L. Yavits, A. Morad, R. Ginosar, "The effect of communication and synchronization on Amdahl's law in multicore systems", Technion TR, http://webee.technion.ac.il/publication-link/index/id/611.

[21] L. Zhao and R. Iyer, *et al*., "Performance, Area, and Bandwidth Implications for Large scale CMP Cache Design", CMP-MSI, 2007.

[22] M. D. Hill and M. R. Marty. "Amdahl's Law in the Multicore Era," *IEEE Computer*, July 2008.

[23] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.0 beta. http://cvxr.com/cvx, September 2013.

[24] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs, Recent Advances in Learning and Control (a tribute to M. Vidyasagar), V. Blondel, S. Boyd, and H. Kimura, editors, pages 95-110, *Lecture Notes in Control and Information Sciences*, Springer, 2008. http://stanford.edu/~boyd/graph_dcp.html.

[25] M. Grant, S. Boyd, Y. Ye. "CVX: Matlab software for disciplined convex programming", 2008.

[26] Mark Hempstead, Gu-Yeon Wei, and David Brooks. "Navigo: An early-stage model to study power-constrained architectures and specialization," ISCA Workshop on Modeling, Benchmarking, and Simulations (MoBS). Austin TX., June 2009

[27] MATLAB Optimization Toolbox Release 2012b, The MathWorks, Inc., Natick, Massachusetts, United States. "lsqcurvefit" solver, http://www.mathworks.com/help/optim/ug/lsqcurvefit.html

[28] N. Hardavellas *et al*., "Toward dark silicon in servers." Micro, IEEE 31.4 (2011): 6-15.

[29] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Cacti 6.0: A tool to understand large caches." University of Utah and Hewlett Packard Laboratories, Tech. Rep (2009).

[30] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0", MICRO, pp. 3–14, 2007.

[31] R. T. Rockefellar. "Lagrange multipliers and optimality", *SIAM Review*, vol. 35, pp. 183–283, 1993.

[32] Rotem, E.; Ginosar, R.; Weiser, U.; Mendelson, A., "Energy Aware Race to Halt: A Down to EARtH Approach for Platform Energy Management," *Computer Architecture Letters*, vol.PP, no.99, pp.1,1, 0. doi: 10.1109/L-CA.2012.32

[33] S. Borkar. "Thousand Core Chips: A Technology Perspective," *Proc. ACM/IEEE 44th Design Automation Conf. (DAC)*, 2007, pp. 746-749.

[34] S. Boyd, L. Vandenberghe, "Convex optimization", Cambridge university press, 2004.

[35] S. Natarajan, S. Narayanan, B. Swamy, and A. Seznec. "Modeling multi-threaded programs execution time in the many-core era." (2013).

[36] S. Wilton and N. Jouppi, "An Enhanced Access and Cycle Time Model for On-Chip Caches ", WRL Research Report 93/5, 1994.

[37] T. Oh *et al*., "An Analytical Model to Study Optimal Area Breakdown between Cores and Caches in a Chip Multiprocessor," IEEE Computer Society Annual Symposium on VLSI, 2009, pp. 181–186.

[38] T. Wada, S. Rajan, , S. Przybylski, "An Analytical Access Time Model for On-Chip Cache Memories", IEEE Journal of solid-state circuits, vol. 27. no. 8, August 1992.

[39] T. Y. Morad, U. C. Weiser, A. Kolodny, M. Valero and E. Ayguadé. "Performance, power efficiency, and scalability of asymmetric cluster chip multiprocessors," *IEEE Computer Architecture Letters*, vol. 4, 2005.

[40] T. Zidenberg, I. Keslassy and U. C. Weiser. "MultiAmdahl: How Should I Divide My Heterogeneous Chip?," *IEEE Computer Architecture Letters*, 2012.

[41] W. Liwei *et al.* "Application specific buffer allocation for wormhole routing Networks-on-Chip", Network on Chip Architectures, 2008, 37.

[42] Y. Ben-Itzhak, I. Cidon, A. Kolodny. "Delay analysis of wormhole based heterogeneous NoC", Fifth IEEE/ACM International Symposium on Networks on Chip, 2011.

[43] Y. Tsai, Y. Xie, V. Narayanan, and M. J. Irwin, "Three-Dimensional Cache Design Exploration Using 3DCacti", ICCD, pp. 519-524, 2005.

[44] Yongpan Liu *et al.*, "Accurate Temperature-Dependent Integrated Circuit Leakage Power Estimation is Easy," *Design, Automation & Test in Europe Conference & Exhibition*, 2007. DATE '07 , vol., no., pp.1,6, 16-20 April 2007, doi: 10.1109/DATE.2007.364517

[45] Z. Guz, Keidar, Idit, Avinoam Kolodny, and Uri C. Weiser. "Nahalal: Memory Organization for Chip Multiprocessors." *Technion-IIT, Department of Electrical Engineering*, 2006.

Amir Morad received his BSc and MSc in Electrical Engineering from the Technion. Amir co-founded VisionTech, a major provider of ICs for set top boxes market. Following VisionTech's acquisition by Broadcom, Amir co-founded Horizon Semiconductors, where he co-designed SoCs for HD cable and satellite set top boxes.

Amir is a PhD student in Electrical Engineering in the Technion. He co-authored a number of patents and research papers on SoC and ASICs. His research interests include analytical modeling and optimization of manycore architectures.

Leonid Yavits received his MSc in Electrical Engineering from the Technion. After graduating, he co-founded VisionTech where he co-designed a single chip MPEG2 codec. Following VisionTech's acquisition by Broadcom, he co-founded Horizon Semiconductors where he co-designed a Set Top Box on chip for cable and satellite TV.

Leonid is a PhD student in Electrical Engineering in the Technion. He co-authored a number of patents and research papers on SoC and ASIC. His research interests include Processing in Memory and 3D IC design.

Ran Ginosar received his BSc from the Technion and his PhD from Princeton University. After conducting research at AT&T Bell Laboratories, he joined the Technion where he is now professor at the Electrical Engineering department and a head of the VLSI Research Center.

Professor Ginosar has been a visiting Associate Professor with the University of Utah and co-initiated the Asynchronous Architecture Research Project at Intel (Oregon). He has co-founded a number of VLSI companies. Professor Ginosar has published numerous papers and patents on VLSI. His research interests include VLSI architecture, asynchronous logic and synchronization.